

# Reliable Minimax Parameter Estimation

**Luc Jaulin**

Laboratoire des Signaux et Systèmes, CNRS  
Supélec, Plateau de Moulon,  
91192 Gif-sur-Yvette Cedex, France.

On leave from  
Laboratoire d'Ingénierie des Systèmes Automatisés,  
62 avenue Notre Dame du Lac,  
49000 Angers, France.

Phone: (33) 1 69 85 17 59  
Fax: (33) 1 69 41 30 60  
Email: jaulin@lss.supelec.fr

**Abstract:** This paper deals with the minimax parameter estimation of nonlinear parametric models from experimental data. Taking advantage of the special structure of the minimax problem, a new efficient and reliable algorithm based on interval constraint propagation is proposed. As an illustration, the ill-conditioned problem of estimating the parameters of a two-exponential model is considered.

## 1 Introduction

This paper deals with estimating the unknown parameters of a model from given experimental data  $y_k, k \in \{1, \dots, k_{\max}\}$  collected on a given system [16]. The available parametric model structure is denoted by  $\mathcal{M}(\cdot)$ ,  $\vec{p} = (p_1, p_2, \dots, p_n)^T \in \mathcal{R}^n$  is the parameter vector to be estimated and  $\vec{y}$  is the vector of all collected data. Each model  $\mathcal{M}(\vec{p})$  generates a vector output  $\vec{y}_m(\vec{p})$  homogeneous to the data vector  $\vec{y}$ . The parameter vector  $\vec{p}$  is assumed to belong to the *prior* feasible box  $[\vec{p}]$ . Define the error between the model output and the data by  $\vec{f}(\vec{p}) \triangleq \vec{y}_m(\vec{p}) - \vec{y}$ . A minimax approach for this problem aims at minimizing the criterion

$$j(\vec{p}) \triangleq \max_{k \in \{1, \dots, k_{\max}\}} |f_k(\vec{p})|, \quad (1)$$

where  $f_k(\vec{p})$  denotes the  $k$ th coordinate function of  $\vec{f}(\vec{p})$ . The minimum  $j^*$  of  $j(\vec{p})$  is the real number

$$j^* \triangleq \min_{\vec{p} \in [\vec{p}]} j(\vec{p}) = \min_{\vec{p} \in [\vec{p}]} \max_{k \in \{1, \dots, k_{\max}\}} |f_k(\vec{p})|, \quad (2)$$

and the set of minimizers, which is often a singleton, is defined by

$$\mathcal{S}^* \triangleq j^{-1}(j^*) = \arg \min_{\vec{p} \in [\vec{p}]} j(\vec{p}). \quad (3)$$

This problem is known as a *discrete Chebyshev problem* and can be regarded as a special case of the general discrete minimax problem. Applications can be found for instance in *sensor fusion* [12], [13] or in *decision theory* [4] where one should minimize the maximum probability of unacceptable error or risk. Let us illustrate this problem by a simple example which will be used later in the paper.

**Example 1** *The problem of finding the smallest disk  $\mathcal{D}$  which contains  $n$  points  $A_1, \dots, A_n$  of  $\mathcal{R}^2$  is a minimax problem. It is trivial to show that the center of the solution disk is the minimizer of the criterion*

$$j(p_1, p_2) = \max_{k \in \{1, \dots, n\}} \left\{ \sqrt{(p_1 - x_{A_k})^2 + (p_2 - y_{A_k})^2} \right\} \quad (4)$$

and its radius is the minimum  $j^*$ . If, for instance,  $n = 3$  and the three points are given by  $A_1(0, 4)$ ,  $A_2(0, -4)$ ,  $A_3(4, 0)$ , the minimizer is  $p^* = (0, 0)^T$  and the minimum is  $j^* = 4$ . Note that  $j$  is not differentiable in  $p^*$ . Figure 1 gives a representation of the isocriteria of  $j(\mathbf{p})$ .  $\diamond$

Since  $j(\vec{p})$  may be non differentiable, because of the presence of the max operator in its expression, a traditional gradient type method cannot be applied efficiently. Other existing methods are essentially local and based on successive linear programming or nonlinear programming techniques (see *e.g.* [18]). In this paper, a global and guaranteed approach will be considered to solve the discrete Chebyshev problem. To the best of my knowledge, such a reliable approach has only been considered for monodimensional problems ( $\dim \mathbf{p} = 1$ ) in [19]. The proposed technique is based on interval constraint propagation and will be shown to solve efficiently ill-conditioned minimax estimation problems in a reliable way.

The paper is organized as follows. Section 2 presents the notion of Cartesian domain for representing sets. Section 3 introduces a special class of sets, namely the *tree sets*, for which the smallest outer Cartesian domain can be computed easily. A new algorithm for characterizing the solution set  $\mathcal{S}^*$  is given in Section 4. This algorithm uses interval constraints propagation and the notions presented in Sections 2 and 3. Section 5 illustrates the efficiency of the algorithm on the minimax estimation of a two-exponential model and a comparison with some classical local minimization methods is given. A notation table can be found at the end of the paper.

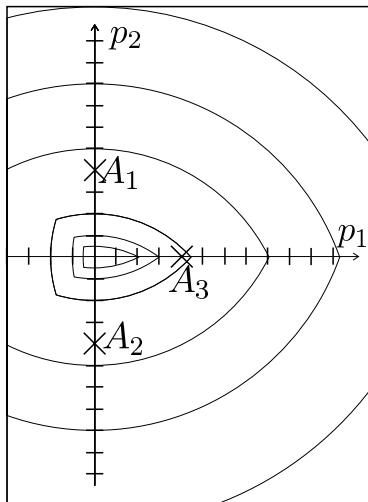


Figure 1: Isocriteria of the criterion involved in Example 1

## 2 Cartesian domains

A *domain* of  $\mathcal{R}$  is a union of intervals of  $\mathcal{R}$ . Interval computation can easily be extended to domains (see, *e.g.*, [10], where domains are called *tolerances*).

When dealing with intervals, two different types of pessimisms may appear during the computation:

- The *dependency pessimism* [14]. It is introduced when a variable occurs more than once in the expression to be evaluated. It can be reduced, for instance, by using the centered form.
- The *hull pessimism*. It is due to the fact that non continuous or multiform<sup>1</sup> functions may be involved. The hull pessimism is illustrated by the two following examples and can be eliminated by using domains instead of intervals.

**Example 2** : Define the sign function:  $\text{sign}(x) = 1$  if  $x \geq 0$  and  $-1$  otherwise. Consider the continuous function  $f(x) = \text{sqr}(\text{sign}(x))$ , where  $\text{sqr}$  denotes the square function. Since  $x$  occurs only once in the expression of  $f(x)$ , no dependency pessimism exists. The image by  $f$  of the interval  $[x] = [-1, 1]$  is  $f([-1, 1]) = 1$ . Nevertheless, an interval evaluation yields

$$[f_I]([-1, 1]) = \text{sqr}(\text{sign}([-1, 1])) = \text{sqr}([-1, 1]) = [0, 1] \quad (5)$$

---

<sup>1</sup>An element  $x$  by a *multiform function*  $f$  may have more than one image. In this sense,  $f(x)$  can be seen a set-valued function.

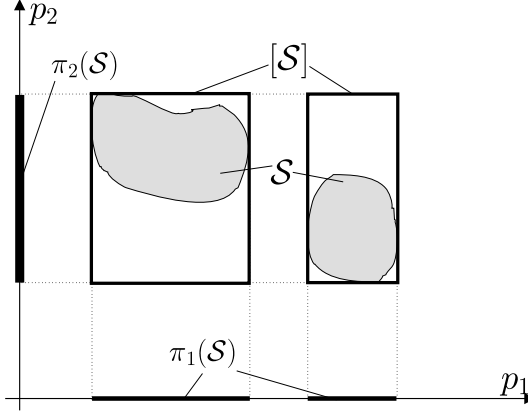


Figure 2: Cartesian hull of a disconnected set  $\mathcal{S}$

and is thus pessimistic. A domain evaluation yields

$$[f_D]([-1, 1]) = \text{sqr}(\text{sign}([-1, 1])) = \text{sqr}(\{-1, 1\}) = \text{sqr}(\{-1\}) \cup \text{sqr}(\{1\}) = \{1\}.$$

which is equal to the image  $f([-1, 1])$ .  $\diamond$

**Example 3 :** The interval propagation approach (see [6],[8]), which will be used in this paper, handle multiform functions such as  $\sin^{-1}$ . Domains have to be used instead of intervals to avoid the hull problem<sup>2</sup>. For instance,  $\sin(\sin^{-1}([0.5, 2])) = [-1, 1]$  if we use interval arithmetic and  $[0.5, 1]$  if we use a domain arithmetic<sup>3</sup>. Note that from a domain point of view,  $\sin^{-1}([0.5, 1])$  is the union of an infinite number of intervals and is usually combined with an intersection with a bounded domain.  $\diamond$

A Cartesian domain of  $\mathcal{R}^n$  is the Cartesian product of  $n$  domains of  $\mathcal{R}$ . The Cartesian hull of a set  $\mathcal{S}$  of  $\mathcal{R}^n$  is the Cartesian product  $[\mathcal{S}]$  of the  $n$  canonical projections  $\pi_i(\mathcal{S})$ ,  $i \in \{1, \dots, n\}$  of  $\mathcal{S}$  onto the  $n$  canonical axis. Cartesian domains will be written with bracketed calligraphic capital letters. On Figure 2, a set  $\mathcal{S} \in \mathcal{R}^2$  with two connected components is represented in grey. The associated Cartesian hull  $[\mathcal{S}]$  is the union of the two boxes represented on the picture. Note that when  $\mathcal{S}$  is connected, its Cartesian hull is the smallest box that contains it.

Let  $\mathcal{A}$ ,  $\mathcal{B}$  and  $\mathcal{C}$  be three subsets of  $\mathcal{R}^n$ . The Cartesian intersection of  $\mathcal{A}$  and  $\mathcal{B}$  is defined by

$$\mathcal{A} \sqcap \mathcal{B} \triangleq [\mathcal{A} \cap \mathcal{B}]. \quad (6)$$

<sup>2</sup> $\sin^{-1}$  has to be understood in a set theoretic sense. In this paper,  $f^{-1}(\mathcal{Y})$  denotes the set  $\{\bar{x} | f(\bar{x}) \in \mathcal{Y}\}$ .

<sup>3</sup>Check that the set property  $f(f^{-1}(\mathcal{Y})) \subset \mathcal{Y}$  is satisfied (since  $[0.5; 1] \subset [0.5; 2]$ ).

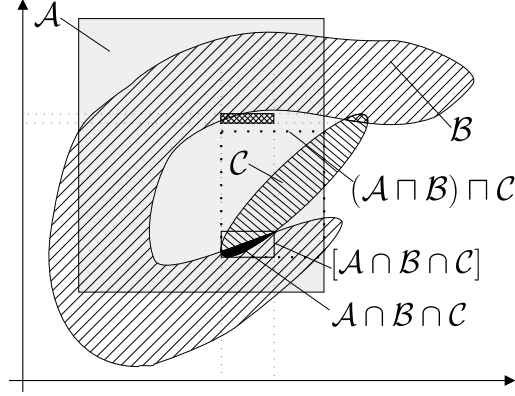


Figure 3: Illustration of the properties of the Cartesian intersection

As we shall see later,  $\mathcal{A} \sqcap \mathcal{B}$  is generally easier to get than  $\mathcal{A} \cap \mathcal{B}$ . The following properties hold:

$$\begin{aligned}
 (i) \quad & \mathcal{A} \subset [\mathcal{A}] && ([\ ] \text{ is pessimistic}) \\
 (ii) \quad & \mathcal{A} \sqcap \mathcal{B} = \mathcal{B} \sqcap \mathcal{A} && (\sqcap \text{ is symmetric}) \\
 (iii) \quad & [\mathcal{A} \cap \mathcal{B} \cap \mathcal{C}] \subset (\mathcal{A} \sqcap \mathcal{B}) \sqcap \mathcal{C} && (\sqcap \text{ is pessimistic}) \\
 (iv) \quad & \mathcal{A} \subset \mathcal{B} \text{ and } \mathcal{A} \subset \mathcal{C} \Rightarrow \mathcal{A} \subset (\mathcal{B} \sqcap \mathcal{C}) && (\text{narrowing property})
 \end{aligned} \tag{7}$$

The proofs of all these properties are trivial. Remark also that  $\sqcap$  is not associative, *i.e.*,  $\exists \mathcal{A} \in \mathcal{R}^n, \exists \mathcal{B} \in \mathcal{R}^n, \exists \mathcal{C} \in \mathcal{R}^n$  such that  $(\mathcal{A} \sqcap \mathcal{B}) \sqcap \mathcal{C} \neq \mathcal{A} \sqcap (\mathcal{B} \sqcap \mathcal{C})$  as illustrated by Figure 3. On this figure,  $\mathcal{A} \sqcap (\mathcal{B} \sqcap \mathcal{C})$  has two disjoint components. One is the box  $[\mathcal{A} \cap \mathcal{B} \cap \mathcal{C}]$  and the other one is the cross-hatched box. Figure 3 illustrates also the property (iii), *i.e.*,  $[\mathcal{A} \cap \mathcal{B} \cap \mathcal{C}] \subset (\mathcal{A} \sqcap \mathcal{B}) \sqcap \mathcal{C}$ . For simplicity of notation, a chain of Cartesian intersections, that has to be evaluated from the left to the right, will be written without parenthesis. For instance,  $\mathcal{A} \sqcap \mathcal{B} \sqcap \mathcal{C}$  means  $(\mathcal{A} \sqcap \mathcal{B}) \sqcap \mathcal{C}$ .

**Remark 1** To find the Cartesian domain  $[\mathcal{S}]$  of  $\mathcal{S} = \mathcal{A} \cap \mathcal{B} \cap \mathcal{C}$ , by using only the operator  $\sqcap$ , a classical idea coming from constraint propagation [17] can be used: compute the following sequence of Cartesian domains:  $[\mathcal{S}](1) = \mathcal{R}^n \sqcap \mathcal{A}$ ,  $[\mathcal{S}](2) = [\mathcal{S}](1) \sqcap \mathcal{B}$ ,  $[\mathcal{S}](3) = [\mathcal{S}](2) \sqcap \mathcal{C}$ ,  $[\mathcal{S}](4) = [\mathcal{S}](3) \sqcap \mathcal{A}$ ,  $\dots$ . Because of the narrowing property (see (iv), formulae (7)), the sequence  $[\mathcal{S}](k)$  is always an enclosure of  $\mathcal{S}$  and converges (sometimes in a finite number of steps) to a Cartesian domain which contains (sometimes is equal to)  $[\mathcal{S}]$ . In our example,  $[\mathcal{S}](5) = \mathcal{A} \sqcap \mathcal{B} \sqcap \mathcal{C} \sqcap \mathcal{A} \sqcap \mathcal{B}$  is equal to  $[\mathcal{S}]$ , but often, the loop converges to a Cartesian domain larger than  $[\mathcal{S}]$ .  $\diamond$

In the following section, we present a special class of sets for which the Cartesian hull can be computed easily.

### 3 Cartesian hull of tree sets

A function  $f$  is *tree decomposable* (or is a *tree function* for short) if (i) in the expression of  $f$  each variable  $p_1, \dots, p_n$  appears at most once and (ii) only the operators  $+, *, -, /$ , and elementary functions like  $\exp, \sin, \cos, \text{sqr}, \dots$  are involved. For instance, the function  $f(\vec{p}) \triangleq p_1 \exp(p_2) + 5 \sin(p_3)$  is a tree function. A tree function can be represented by a syntactic tree where the nodes are operators or elementary functions and the leaves are either one of the  $p_i$ 's or a constant number. Tree functions are also known in the literature as functions with a totally decomposable tree structure decomposition (or TDTSD function [1], [2]). A *tree set* is a set which can be written as

$$\mathcal{S} = [\mathcal{P}] \cap f^{-1}(\mathcal{Y}), \quad (8)$$

where (i)  $f : \mathcal{R}^n \rightarrow \mathcal{R}; \vec{p} \rightarrow f(\vec{p})$  is a tree function, (ii)  $[\mathcal{P}]$  is a Cartesian domain and (iii)  $\mathcal{Y}$  is a domain of  $\mathcal{R}$ .

Interval techniques make it possible to obtain the Cartesian hull of a tree set in a very efficient way. Consider, for instance, the set  $\mathcal{S}$  given by (8) where  $f(\vec{p}) \triangleq p_1 \exp(p_2) + 5 \sin(p_3)$ . To obtain the projections of  $\mathcal{S}$  on the coordinate axis (see, e.g., [3]), it suffices to decompose the relation between  $y$  and the  $p_i$ 's into primitive constraints as follows

$$y = p_1 \exp(p_2) + 5 \sin(p_3) \Leftrightarrow \begin{cases} y = x_1 + x_2 \\ x_1 = p_1 x_3 \\ x_3 = \exp(p_2) \\ x_2 = 5x_4 \\ x_4 = \sin(p_3) \end{cases} \quad (9)$$

where the  $x_i$ 's are intermediate variables. Note that each  $x_i$  appears exactly once on the left and once on the right. Each  $p_i$  appears exactly once on the right and  $y$  appears exactly once on the left. To compute, for example, the third component  $\mathcal{S}_3$  of  $[\mathcal{S}]$ , transform the set of primitive constraints in order to have  $y$  on the right and  $p_3$  on the left:

$$\begin{cases} x_2 = y - x_1 \\ x_1 = p_1 x_3 \\ x_3 = \exp(p_2) \\ x_4 = x_2/5 \\ p_3 = \sin^{-1}(x_4) \end{cases} \Leftrightarrow p_3 = \sin^{-1} \left( \frac{y - p_1 \exp(p_2)}{5} \right). \quad (10)$$

Then

$$\mathcal{S}_3 = \pi_3([\mathcal{P}] \cap f^{-1}(\mathcal{Y})) = \mathcal{P}_3 \cap \sin^{-1} \left( \frac{\mathcal{Y} - \mathcal{P}_1 * \exp(\mathcal{P}_2)}{5} \right), \quad (11)$$

where  $\pi_3$  is the projection operator onto the third axis. Using domains instead of intervals for representing the possible values for the variables handled makes it possible to obtain the exact projections of the tree set and not only an enclosure, *i.e.*, the domain arithmetic is not hull pessimistic. The following example illustrates this point.

**Example 4** Consider the tree set

$$\mathcal{S} = \{p \in \mathcal{P} = [-\pi, \pi] \mid \text{sqr}(\sin(p)) \in [1/2, 2]\}, \quad (12)$$

where  $\text{sqr}$  is the square function. The decomposition into primitive constraints is

$$y = \text{sqr}(\sin(p)) \Leftrightarrow \begin{cases} y = \text{sqr}(x_1) \\ x_1 = \sin(p) \end{cases} \Leftrightarrow \begin{cases} x_1 = \text{sqr}^{-1}(y) \\ p = \sin^{-1}(x_1) \end{cases}$$

Let us stress once more that  $\text{sqr}^{-1}(y)$  and  $\sin^{-1}(x_1)$  have to be interpreted in a set theoretic sense and should not be confused with  $\sqrt{y}$  and  $\arcsin(x_1)$ . For instance,  $\sqrt{4} = 2$  whereas  $\text{sqr}^{-1}(4) = \{-2, 2\}$ . We get

$$\mathcal{X}_1 = \text{sqr}^{-1}(\mathcal{Y}) = \text{sqr}^{-1}([1/2, 2]) = [-\sqrt{2}, -1/\sqrt{2}] \cup [1/\sqrt{2}, \sqrt{2}], \quad (13)$$

$$\mathcal{S} = \sin^{-1}(\mathcal{X}_1) \cap [-\pi, \pi] \quad (14)$$

$$= \sin^{-1}(\left([- \sqrt{2}, -1/\sqrt{2}] \cup [1/\sqrt{2}, \sqrt{2}]\right)) \cap [-\pi, \pi] \quad (15)$$

$$= [-3\pi/4, -\pi/4] \cup [\pi/4, 3\pi/4]. \quad (16)$$

Using intervals instead of domains gives

$$\mathcal{X}_1 = \text{sqr}^{-1}(\mathcal{Y}) = \text{sqr}^{-1}([1/2, 2]) = [-\sqrt{2}, \sqrt{2}], \quad (17)$$

$$\mathcal{P} = \sin^{-1}(\mathcal{X}_1) \cap [-\pi, \pi] = [-\pi, \pi], \quad (18)$$

which contains  $\mathcal{S}$  but  $[-\pi, \pi]$  is not the smallest interval containing  $\mathcal{S}$  (which is  $[-3\pi/4, 3\pi/4]$ ). $\diamond$

**Remark 2** This section has presented a method to compute efficiently the Cartesian hull of a tree set. In general, the Cartesian intersection of two tree sets is not easy to obtain. Nevertheless, computing the Cartesian intersection between a Cartesian domain  $[\mathcal{Q}]$  and a tree set  $\mathcal{S} = [\mathcal{P}] \cap f^{-1}(\mathcal{Y})$  amounts to finding the Cartesian hull of the tree set  $([\mathcal{Q}] \cap [\mathcal{P}]) \cap f^{-1}(\mathcal{Y})$ . The reason is that

$$[\mathcal{Q}] \cap ([\mathcal{P}] \cap f^{-1}(\mathcal{Y})) = [[\mathcal{Q}] \cap [\mathcal{P}] \cap f^{-1}(\mathcal{Y})] = [([\mathcal{Q}] \cap [\mathcal{P}]) \cap f^{-1}(\mathcal{Y})]. \quad (19)$$

Therefore, in the procedures, presented in the next section, only Cartesian intersections between Cartesian domains and tree sets will be allowed.  $\diamond$

## 4 Minimax algorithm

This section presents the new algorithm MINIMAX. This algorithm, presented on Table 1, generates a list of boxes  $\mathcal{S}^+$  that contains the set of all global minimizers  $\mathcal{S}^*$  given by (3) and computes the optimum  $j^*$  given by (2). All  $f_k$ 's are assumed to be tree functions.  $[\vec{p}_0]$  is the initial search box, assumed large enough to contain  $\mathcal{S}^*$ .  $Q$  is a First-In-First-Out list of boxes. MINIMAX first calls a local minimization procedure CROSS at Step 3 to decrease the upper bound  $j^+$  for  $j^*$ .  $j^+$  is a global variable of all procedures. The procedure NARROW, called at Step 4, attempts to reduce the current box  $[\vec{p}]$  without losing any global minimizer, *i.e.*, the resulting box  $[\vec{q}]$  satisfies  $[\vec{q}] \cap \mathcal{S}^* = [\vec{p}] \cap \mathcal{S}^*$ . The algorithm CROSS and two versions of NARROW, based on the notions of Cartesian intersections and interval constraint propagation, are given in the following subsections.

- MINIMAX( $[\vec{p}_0]$ )
- 1      $Q := \{[\vec{p}_0]\}; \mathcal{S}^+ := \emptyset, j^+ := \infty;$
  - 2     if  $Q \neq \emptyset$ , put the first element of  $Q$  into  $[\vec{p}]$  else go to 8;
  - 3      $j^+ := \text{CROSS}(\text{Center}([\vec{p}]), j^+);$
  - 4      $[\vec{q}] := \text{NARROW}([\vec{p}], j^+);$
  - 5     If  $[\vec{q}] = \emptyset$ , go to 2;
  - 6     If  $\text{width}([\vec{q}]) \leq \varepsilon$ ,  $\{\mathcal{S}^+ := \mathcal{S}^+ \cup \{[\vec{q}]\}; \text{Go to 2};\}$
  - 7     Bisect  $[\vec{q}]$  and store the two resulting boxes at the end of  $Q$ . Go to 2;
  - 8     Remove all boxes  $[\vec{q}]$  of  $\mathcal{S}^+$  that are such that  $j([\vec{q}]) > j^+$ .

Table 1: Algorithm that minimizes the criterion (1) inside the box  $[\vec{p}_0]$

**Remark 3** *In this paper, local may have two different meanings. We distinguish the locality with respect to the constraints (c-locality) and the locality with respect to the parameter space (s-locality). For instance, we shall say that  $p^*$  is a s-local minimizer if there exists a neighborhood  $\mathcal{N}$  of  $p^*$  such that  $\forall \vec{p} \in \mathcal{N}, j(\vec{p}) \geq j(p^*)$ . When dealing with a constraint satisfaction problem (CSP, see, e.g., [3]), we shall speak about a c-local approach when the constraints are considered one by one and a c-global approach when all constraints are considered together. For instance, Remark 1 in Section 2 presented a c-local procedure to get an outer approximation of the Cartesian hull of the intersection of three sets.  $\diamond$*

### 4.1 The s-local research algorithm CROSS

An iterative algorithm to minimize the criterion (1) (s-locally), as required by Step 3 of MINIMAX, is now given. This algorithm has been presented for the first time in [11]. Let  $\vec{p}$  be

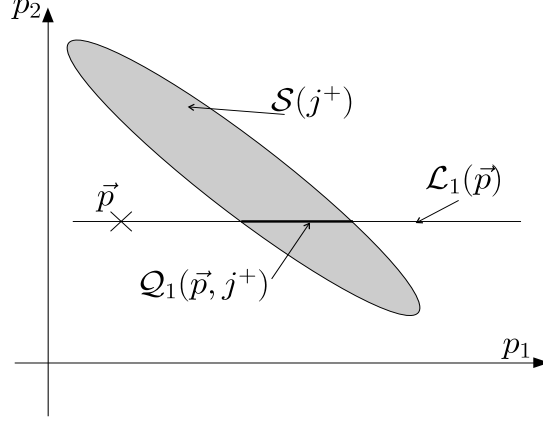


Figure 4: Local research from  $\vec{p}$  along the first direction

a parameter vector,  $i$  a given direction in the parameter space and  $j^+$  a real number which satisfies  $j^* \leq j^+ \leq j(\vec{p})$  (if such a  $j^+$  is not available, take  $j^+ = j(\vec{p})$ ). In order to perform one  $s$ -local minimization iteration from  $\vec{p}$  along the direction  $i$ , define the following sets:

$$\mathcal{L}_i(\vec{p}) \triangleq \{\vec{q} \in \mathcal{R}^n | \exists q_i \in \mathcal{R}, \vec{q} = (p_1, \dots, p_{i-1}, q_i, p_{i+1}, \dots, p_n)\}, \quad (20)$$

$$\mathcal{S}_k(j^+) \triangleq \{\vec{p} | |f_k(\vec{p})| \leq j^+\} = f_k^{-1}([-j^+, j^+]), \quad (21)$$

$$\mathcal{S}(j^+) \triangleq \bigcap_{k=1}^{k_{\max}} \mathcal{S}_k(j^+) = \{\vec{p} \in \mathcal{R}^n | \forall k, |f_k(\vec{p})| \leq j^+\} = \{\vec{p} \in \mathcal{R}^n | j(\vec{p}) \leq j^+\}, \quad (22)$$

$$\mathcal{Q}_i(\vec{p}, j^+) \triangleq \mathcal{L}_i(\vec{p}) \cap \mathcal{S}(j^+) = \mathcal{L}_i(\vec{p}) \cap \mathcal{S}_1(j^+) \cap \dots \cap \mathcal{S}_{k_{\max}}(j^+). \quad (23)$$

A representation of the sets  $\mathcal{L}_i(\vec{p})$ ,  $\mathcal{S}(j^+)$ ,  $\mathcal{Q}_i(\vec{p}, j^+)$  is given in Figure 4, in a 2 dimensional context and for  $i = 1$ . Since  $\mathcal{L}_i(\vec{p})$  is a line parallel to the  $i$ th axis, it is trivial to prove that

$$\mathcal{Q}_i(\vec{p}, j^+) \triangleq \mathcal{L}_i(\vec{p}) \cap \mathcal{S}_1(j^+) \cap \dots \cap \mathcal{S}_{k_{\max}}(j^+) = \mathcal{L}_i(\vec{p}) \cap \mathcal{S}(j^+). \quad (24)$$

Therefore, when the  $f_k$ 's are tree functions,  $\mathcal{Q}_i(\vec{p}, j^+)$  is easily obtained by computing  $k_{\max}$  Cartesian hulls of  $k_{\max}$  tree sets (see Remark 2 of Section 3). Moreover, any point  $\vec{q}$  inside  $\mathcal{Q}_i(\vec{p}, j^+)$  satisfies  $j(\vec{q}) \leq j^+$  and can thus be used to decrease the upper bound  $j^+$  along the direction  $i$ .

The algorithm presented on Table 2 takes advantage of this idea to decrease the upper bound  $j^+$  for  $j^*$ .  $\kappa > 0$  is a small real number used to stop the procedure when the improvement is not significant enough.

CROSS( $\vec{p}, j^+$ )

- 1  $\hat{j} := j^+$ ;
- 2 For all  $i \in \{1, \dots, n\}$
- 3      $\mathcal{Q}_i(\vec{p}, j^+) := \mathcal{L}_i(\vec{p}) \cap \mathcal{S}(j^+)$ ;
- 4     If  $\mathcal{Q}_i(\vec{p}, j^+) = \emptyset$ , next  $i$ ;
- 5     Select a point  $\vec{q}$  inside  $\mathcal{Q}_i(\vec{p}, j^+)$ ;
- 6     If  $j(\vec{q}) \leq \hat{j}$ ,  $\hat{q} := \vec{q}$ ,  $\hat{j} := j(\vec{q})$ ;
- 7 EndFor
- 8 if  $j^+ - \hat{j} > \kappa$ ,  $\{j^+ := \hat{j}; \vec{p} := \hat{q}; \text{Go to 2};\}$
- 9 return  $j^+$ .

Table 2: A  $s$ -local algorithm to decrease the upper bound  $j^+$  of  $j^*$

**Comments:** Except for atypical situations, the set  $\mathcal{Q}_i(\vec{p}, j^+)$ , computed at Step 3, is a segment or a finite union of aligned segments. The center of the largest segment of  $\mathcal{Q}_i(\vec{p}, j^+)$  is generally chosen at Step 5. The situation  $\mathcal{Q}_i(\vec{p}, j^+) = \emptyset$  (Step 4) can only happen when  $j(\vec{p}) > j^+$ , i.e., when the loop is run for the first time. If the improvement on the upper bound is sufficient ( $j^+ - \hat{j} > \kappa$ ), the loop is run again from  $\hat{q}$ .  $\diamond$

**Example 5** Run CROSS on Example 1 with  $\vec{p} = (2, 8)$  and  $j^+ = 6$ . The three sets  $\mathcal{S}_1(j^+)$ ,  $\mathcal{S}_2(j^+)$ ,  $\mathcal{S}_3(j^+)$  are the lightgrey disks represented on Figure 5. The darkgrey set is  $\mathcal{S}(j^+)$ . The loop is first run for  $i = 1$ , and at Step 3, CROSS computes  $\mathcal{Q}_1(\vec{p}, j^+)$  as follows:

$$\begin{aligned}
\mathcal{Q}_1(\vec{p}, j^+) &= \mathcal{L}_1(\vec{p}) \cap \mathcal{S}(j^+) \\
&= \mathcal{L}_1(\vec{p}) \cap \mathcal{S}_1(j^+) \cap \mathcal{S}_2(j^+) \cap \mathcal{S}_3(j^+) \\
&\stackrel{(24)}{=} \{(\mathcal{L}_1(\vec{p}) \cap \mathcal{S}_1(j^+)) \cap \mathcal{S}_2(j^+)\} \cap \mathcal{S}_3(j^+).
\end{aligned} \tag{25}$$

Now,  $\mathcal{L}_1(\vec{p}) \cap \mathcal{S}_1(j^+)$  can be computed by using the method proposed on Section 3:

$$\begin{aligned}
&\sqrt{(p_1 - x_{A_1})^2 + (p_2 - y_{A_1})^2} \in [0, 6] \\
\Leftrightarrow \sqrt{p_1^2 + (p_2 - 4)^2} \in [0, 6] &\Leftrightarrow p_1^2 + (8 - 4)^2 \in [0, 36] \\
\Leftrightarrow p_1^2 + 16 \in [0, 36] &\Leftrightarrow p_1^2 \in [0, 20] \Leftrightarrow p_1 \in [-\sqrt{20}, \sqrt{20}].
\end{aligned} \tag{26}$$

Therefore,  $\mathcal{L}_1(\vec{p}) \cap \mathcal{S}_1(j^+) = [-\sqrt{20}, \sqrt{20}] \times [8, 8]$ . The same reasoning, applied to compute  $(\mathcal{L}_1(\vec{p}) \cap \mathcal{S}_1(j^+)) \cap \mathcal{S}_2(j^+)$ , leads to the empty set. Therefore,  $\mathcal{Q}_1(\vec{p}, j^+) = \emptyset$ . The horizontal direction  $i = 1$  is thus eliminated and the loop is now run for the vertical direction  $i = 2$ . We get

$$\mathcal{Q}_2(\vec{p}, j^+) = \mathcal{L}_2(\vec{p}) \cap \mathcal{S}(j^+) = [2, 2] \times [-1.657, 1.657]. \tag{27}$$

When the loop is left and if the center of  $\mathcal{Q}_2(\vec{p}, j^+)$  is chosen, we get  $\hat{q} = (2 \ 0)^T$  and  $j^+ = \sqrt{20}$ . CROSS is then run again from  $\vec{p} = \hat{q}$ .  $\diamond$

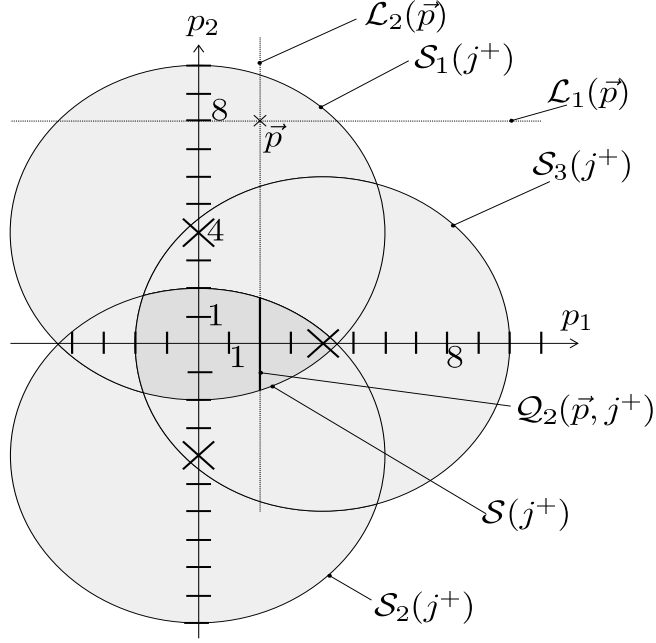


Figure 5: An iteration of the cross algorithm

## 4.2 A c-local narrowing procedure

Given a box  $[\vec{p}]$ , the procedure, to be presented now, computes a subbox  $[\vec{z}]$  of  $[\vec{p}]$  as small as possible such that  $[\vec{z}] \supset [\vec{p}] \cap \mathcal{S}(j^+)$ , as required by Step 4 of MINIMAX. An idealized algorithm would be to compute the Cartesian domain of the set

$$[\vec{p}] \cap \mathcal{S}(j^+) = [\vec{p}] \cap \mathcal{S}_1(j^+) \cap \mathcal{S}_2(j^+) \cap \dots \cap \mathcal{S}_{k_{\max}}(j^+). \quad (28)$$

Unfortunately, this operation is generally impossible to perform directly. The idea of a c-local narrowing algorithm, given in Table 3, is to take the constraints one by one. We compute  $[\mathcal{P}](1) = [\vec{p}] \sqcap \mathcal{S}_1(j^+)$ ,  $[\mathcal{P}](2) = [\mathcal{P}](1) \sqcap \mathcal{S}_2(j^+)$ ,  $\dots$ ,  $[\mathcal{P}](k_{\max}) = [\mathcal{P}](k_{\max} - 1) \sqcap \mathcal{S}_{k_{\max}}(j^+)$ . Then, we loop until we are unable to narrow the enclosure significantly. In Table 3,  $h_{\infty}([\mathcal{Q}], [\mathcal{P}])$  is the Hausdorff distance between sets  $[\mathcal{Q}]$  and  $[\mathcal{P}]$  associated with the  $L_{\infty}$ -norm and quantifies the reduction. If the reduction is smaller than a given real number  $\eta > 0$ , we do not try to narrow the current Cartesian domain  $[\mathcal{P}]$  once more. Because  $\sqcap$  is pessimistic (see formula (7)), the current  $[\mathcal{P}]$  is only a superset of  $[\vec{p}] \cap \mathcal{S}(j^+)$ .

	$\text{C-LOCALNARROW}([\bar{p}], j^+)$
1	$[\mathcal{P}] := [\bar{p}];$
2	$[\mathcal{Q}] := [\mathcal{P}];$
3	For all $k \in \{1, \dots, k_{\max}\}$ , $[\mathcal{P}] := [\mathcal{P}] \cap \mathcal{S}_k(j^+);$
4	If $h_\infty([\mathcal{Q}], [\mathcal{P}]) \geq \eta$ , go to 2;
5	Return the smallest box which contains $[\mathcal{P}];$

*Table 3: A c-local interval-constraint-propagation algorithm*

**Example 6** Let us run C-LOCALNARROW on Example 1 with  $[\bar{p}] = [0, 11] \times [-8, 8]$ . Step 3 generates three boxes  $[\mathcal{P}](k)$ ,  $k \in \{1, 2, 3\}$  such that  $[\mathcal{P}](3) \subset [\mathcal{P}](2) \subset [\mathcal{P}](1) \subset [\mathcal{P}]$  as illustrated by Figure 6. Boxes are shown by two of their extreme vertices. Note that in general, the  $[\mathcal{P}](k)$ 's are not boxes but Cartesian domains.  $\diamond$

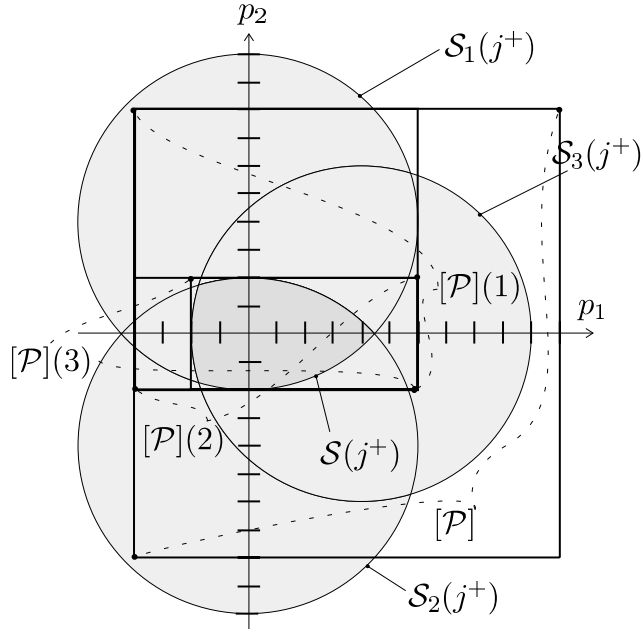


Figure 6: Illustration of the c-local narrowing procedure

### 4.3 A c-global narrowing procedure

This subsection presents a c-global approach to narrow the current box at Step 4 of the algorithm MINIMAX. Thus, all constraints are now considered together and not one by one. The main motivation is that when dealing with a system of  $n$  equations of  $n$  unknown variables, the interval Newton method ([14], [9]) can narrow  $[\bar{p}]$  in a very efficient way. Unfortunately, it

cannot be used when the number of equations is higher than the number of unknown variables (generally the case when dealing with estimation problems).

The approach considered here consists of linearizing the system of nonlinear equations into a system of linear interval equations, the solution set of which encloses the solution set of the initial nonlinear system. The principle of the interval linearization considered here is to bracket the function  $\vec{f}(\vec{p})$ , inside  $[\vec{p}]$ , by two parallel affine functions

$$A.\vec{p} + \vec{b}^- \leq \vec{f}(\vec{p}) \leq A.\vec{p} + \vec{b}^+. \quad (29)$$

To find such an affine enclosure of  $\vec{f}$  over  $[\vec{p}]$ , one can use the mean-value theorem:

$$\forall \vec{p} \in [\vec{p}], \vec{f}(\vec{p}) \in \vec{f}(\vec{p}_0) + \left[ \frac{d\vec{f}}{d\vec{p}} \right]([\vec{p}])(\vec{p} - \vec{p}_0) \text{ with } \vec{p}_0 = \text{center}([\vec{p}]). \quad (30)$$

Then, it can easily be shown that  $\vec{f}(\vec{p}) \in A.\vec{p} + [\vec{b}]$  where

$$A = \left( \frac{d\vec{f}}{d\vec{p}}(\vec{p}_0) \right), \quad (31)$$

$$[\vec{b}] = \vec{f}(\vec{p}_0) - \frac{d\vec{f}}{d\vec{p}}(\vec{p}_0).\vec{p}_0 + \left( \left[ \frac{d\vec{f}}{d\vec{p}} \right]([\vec{p}]) - \frac{d\vec{f}}{d\vec{p}}(\vec{p}_0) \right) ([\vec{p}] - \vec{p}_0). \quad (32)$$

Figure 7 shows an affine enclosure  $a.p + [b] = \frac{1}{4}p + [\frac{1}{3}, 1]$  of a real function  $f$  over  $[p] = [0, 2]$ .

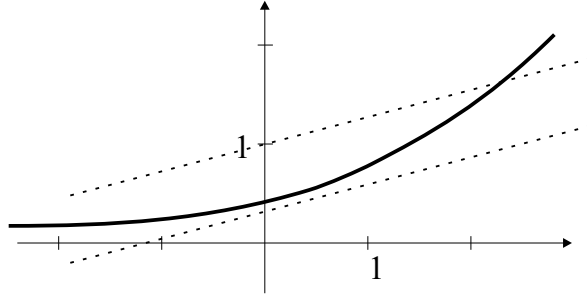


Figure 7: Affine enclosure of a real function

Denote by  $\vec{j}^+$  the vector whose components are all equal to  $j^+$ . Since

$$\vec{p} \in \mathcal{S}(j^+) \Leftrightarrow \vec{f}(\vec{p}) \in [-\vec{j}^+, \vec{j}^+] \Leftrightarrow \exists \bar{j} \in [-\vec{j}^+, \vec{j}^+] | \vec{f}(\vec{p}) = \bar{j} \quad (33)$$

$$\Rightarrow \exists \bar{b} \in [\vec{b}], \exists \bar{j} \in [-\vec{j}^+, \vec{j}^+] | A.\vec{p} + \bar{b} = \bar{j} \quad (34)$$

$$\Leftrightarrow \exists \bar{b} \in [\vec{b}], \exists \bar{j} \in [-\vec{j}^+, \vec{j}^+] | A.\vec{p} = \bar{j} - \bar{b} \quad (35)$$

$$\Leftrightarrow A.\vec{p} \in [-\vec{j}^+, \vec{j}^+] - [\vec{b}] = [-\vec{j}^+ - \vec{b}^+, \vec{j}^+ - \vec{b}^-], \quad (36)$$

a vector  $\vec{p}$  of  $\mathcal{S}(j^+)$ , is necessarily a solution of the system of interval linear equations:

$$A.\vec{p} = \left[ -\vec{j}^+ - \vec{b}^+, \vec{j}^+ - \vec{b}^- \right] \quad (37)$$

which is in general overdetermined. The smallest cube  $[\vec{z}]$  containing the solution of (37) can be computed by linear programming (see also [15]). The box  $[\vec{p}] \cap [\vec{z}]$  encloses the set  $[\vec{p}] \cap \mathcal{S}(j^+)$  as required by Step 4 of MINIMAX.

## 5 Application

This section illustrates on a two-exponential estimation problem the superiority of our approach over classical methods. All computations have been performed on a Pentium 133MHz. Consider a model where the relation between the parameter vector  $\vec{p}$  and the model output is given by

$$y_m(\vec{p}, t) = p_1 \exp(p_2 t) + p_3 \exp(p_4 t). \quad (38)$$

Note that, since a permutation of  $p_1$  with  $p_3$  and of  $p_2$  with  $p_4$  does not affect the model output, the model is not globally identifiable (see [16]). Therefore, any reliable identification method should lead to symmetrical solutions. Assume that 10 data points  $y(1), \dots, y(10)$  have been generated as follows

$$\begin{aligned} y(k) &= 20 \exp(-0.8 t_k) - 10 \exp(-0.2 t_k) + n(k), \\ n(k) &= 0.1 \sin(k), && \text{(unknown noise)} \\ t_k &= \frac{1}{4} k^2, k \in \{1, \dots, 10\}. && \text{(known)} \end{aligned} \quad (39)$$

These data are displayed on Figure 8.

When the initial vector and the options are well chosen, the function `leastsq` of the toolbox `optim` of Matlab finds in 1.2 seconds a local solution vector of the least square criterion and the local procedure `minimax` of the toolbox `optim` [5], finds in about 5 seconds a solution equal to 0.0657. These two Matlab procedures are local, diverge for some bad initial vectors, are sensitive with respect to the initial vector, do not provide any guaranty on their results (even s-locally), often stop because of some ill-conditioning and never detect that the problem has two solutions. By contrast, the approach advocated here is able to solve s-globally and efficiently the minimax problem. For  $\varepsilon = 0.05$  (in MINIMAX),  $\kappa = 0.001$  (in CROSS),  $\eta = 0.001$  (in C-LOCALNARROW) and  $[p_0] = [-60, 60] \times [-1, 0] \times [-60, 60] \times [-1, 0]$ , MINIMAX finds in 1.7 seconds and after 109 bisections the guaranteed enclosure of  $j^* \in [0.0653, 0.0657]$ . The set  $\mathcal{S}^+$  (which encloses  $\mathcal{S}^*$ ) consists of 44 boxes and has two symmetrical disconnected components. If now MINIMAX calls both the c-local narrowing and the c-global narrowing procedures, the

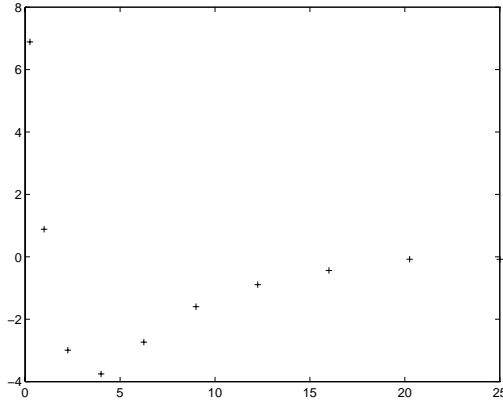


Figure 8: Data associated with the two-exponential test-case

enclosure  $j^* \in [0.0654, 0.06567]$  is found in 1.9 seconds and after 101 bisections. For this test-case, the improvement produced by the c-global narrowing procedure is marginal. Nevertheless, since this procedure makes it possible to avoid some bisections, we can hope that it may decrease the computing time when dealing with problems of higher dimensions or when the nonlinearity is not as important as for our test-case.

## 6 Conclusion

A new algorithm MINIMAX for solving nonlinear discrete minimax problems has been presented. MINIMAX has been shown to be very efficient and leads to guaranteed results. It combines a new local research procedure to decrease the upper bound on the solution with a pruning procedure based on interval constraint propagation. On a two-exponential estimation problem, it has been shown that our reliable algorithm was able to compete (even from a computing-time viewpoint) with optimized Matlab procedures.

In further researches, the MINIMAX algorithm could be improved by using more sophisticated local research algorithm (see [5], [7]) and extended to deal with nontree functions (by decomposition of each constraint into primitive tree constraints). Moreover, the efficiency of the c-global narrowing procedure remains to be studied.

The reason why engineers rely on least-square criteria almost systematically is that for linear models, least-square estimation is equivalent to solving a linear system of equations and by extension, they use least-square criteria even when they deal with nonlinear models. Now, for some class of nonlinear problems, minimax estimation has been shown to be easier to solve in an efficient and reliable way than least-square estimation. As a result, when dealing with nonlinear fitting data problems, without any ideas on the noise distribution, the minimax criterion should

sometime be preferred over classical least-square criteria, because it allows a fast and reliable global optimization.

**Software:** The algorithm MINIMAX has been implemented with Borland-C++ Builder 3.0 to solve the test-case. The source programs and the associated libraries are available on request.

## Notation table

$[\mathcal{S}]$	Cartesian hull of the set $\mathcal{S}$ . See Section 2.
$\mathcal{A} \cap \mathcal{B}$	Cartesian intersection of two sets. See (6).
$j(\vec{p})$	criterion to be minimized. See (1).
$j^*$	minimum value for $j(\vec{p})$ . See (2).
$j^+$	current upper bound for $j^*$ .
$f_k(\vec{p})$	$k$ th error function.
$\vec{p}$	parameter vector to be estimated.
$\mathcal{S}^*$	set of all global minimizers. See (3).
$\mathcal{S}(j^+)$	set of all $\vec{p}$ such that for all $k \in \{1, \dots, k_{\max}\},  f_k(\vec{p})  \leq j^+$ .
$\mathcal{S}^+$	set of boxes which encloses $\mathcal{S}^*$ .
$\vec{y}$	data vector.
$\vec{y}_m(\vec{p})$	vector of model outputs.

## References

- [1] Ackerman J., Barlett A., Kaesbauer D., Siemel W. and Steinhauser R., *Robust Control Systems with Uncertain Physical Parameters*, Springer-Verlag, London, 1993.
- [2] Barmish B., Ackerman J. and Hu H., The Tree Structured Decomposition: a New Approach to Robust Stability Analysis, *Proc. Conf. on Information Sciences and Systems*, Princeton, 1990, 133-139.
- [3] Benhamou F. and Granvilliers L., Automatic Generation of Numerical Redundancies for Nonlinear Constraint Solving, *Reliable Computing*, **3**, 1997, 335-344.
- [4] Berger J., *Statistical Decision Theory and Bayesian Analysis*, New York: Springer-Verlag, second ed., 1985.
- [5] Brayton R.K., Director S.W., Hachtel, G.D. and Vidigal L., A new Algorithm for Statistical Circuit Design Based on Quasi-Newton Methods and Function Splitting, *IEEE Trans. Circuit and Systems*, **26**, 1979, 784-794.
- [6] Cleary J. C., Logical arithmetic, *Future Computing Systems*, **2**(2), 1987, 125-149.
- [7] Conn A. R. and Li Y., A Structure-Exploiting Algorithm for Nonlinear Minimax Problems, *Siam Journal on Optimization*, **2**(2), 1992.
- [8] Davis E., Constraint propagation with interval labels, *Artificial Intelligence*, **32**, 1987, 281-331.
- [9] Hansen E., *Global Optimization Using Interval Analysis*, Marcel Dekker, New York, 1992.
- [10] Hyvönen E., Constraint Reasoning Based on Interval Arithmetic; The Tolerance Propagation Approach, *Artificial Intelligence*, **58**, 1992, 71-112.
- [11] Jaulin L, Interval constraint propagation with application to bounded-error estimation. Submitted to *Automatica*.
- [12] McKendall R., *Minimax Estimation of a Discrete Location Parameter for a Continuous Distribution*, PhD dissertation, Systems Engineering, University of Pennsylvania. Available as technical report MS-CIS-90-28, Computer and Information Science Department, University of Pennsylvania, 1990.
- [13] McKendall R. and Mintz M., Robust Sensor Fusion with Statistical Decision Theory, Chapter in *Data Fusion in Robotics and Machine Intelligence*, M.A.Abidi and R.C.Gonzalez, editors, Academic Press, Boston, 211-244, 1992.

- [14] Moore R.E., *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, 1979.
- [15] Rohn J., Enclosing solutions of overdetermined systems of linear interval equations, *Reliable Computing*, **2**(2), 1996, 167-171.
- [16] Walter E. and Pronzato L., *Identification of Parametric Models from Experimental Data*, Springer, London, 1997.
- [17] Waltz D. L., Generating semantic descriptions from drawings of scenes with shadows. in: P.H. Winston, editor, *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975, 19-91.
- [18] Watson G.A., The Minimax Solution of an Overdetermined System of Nonlinear Equations, *J. Inst. Math. Appl.*, 1979, 167-180.
- [19] Wolfe M. A., On discrete minimax problems in  $\mathcal{R}$  using interval arithmetic, *Reliable Computing*, **5**(4), 1999, 371-383.